4. Model and results

4.1 Fully Connected Structure

In [ ]:
```
#Import all needed libraries
import os
import numpy as np
from os import listdir
from imageio import imread
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from keras.utils.image_utils import img_to_array

import PIL
import matplotlib.pyplot as plt

from tensorflow import keras
import numpy as np
import pandas as pd
import sklearn as sk
import time
from keras.datasets import mnist
from keras.models import Sequential, load_model
from keras.layers import Dense, Dropout, Flatten
from keras import optimizers
from keras import backend as K
from keras import regularizers
from keras import initializers
import keras as ks
from matplotlib import pyplot as plt
```

In [ ]:
```
#Connected to my google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

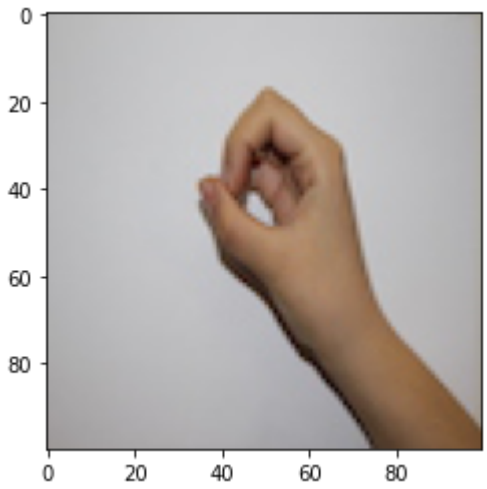
In [ ]:
```
#Settings
num_classes = 10
test_size = 0.2
```

In [ ]:
```
#Read image and convert to 3D array
def get_img(data_path):
  ## Getting image array from path:
  img = PIL.Image.open(data_path)
  img = img.convert("L")
  img = img_to_array(img)
  img = np.resize(img, (100, 100))
#   img = np.load(data_path)
  return img
```

In [ ]:
```
#Get dataset form pictures and split totrain and test sets
from matplotlib import image
from matplotlib import pyplot
#Load image as pixel array
```

```
image = image.imread('/content/drive/MyDrive/4050_Final_Dataset/0/IMG_1118.JPG')
#Summarize shape of the pixel array
print(image.dtype)
print(image.shape)
#Display the array of pixels as an image
pyplot.imshow(image)
pyplot.show()
```

```
uint8
(100, 100, 3)
```



In [ ]:

```
dataset_path = "/content/drive/MyDrive/4050_Final_Dataset"

## Getting all data from data path
labels = sorted(listdir(dataset_path))
print(labels)
X = []
Y = []
for i, label in enumerate(labels):
  data_path = dataset_path + "/" + label

  for data in listdir(data_path):
    img = get_img(data_path + "/" + data)
    X.append(img)
    Y.append(i)
## create dataset
X = 1 - np.array(X).astype("float32") /255
# X = np.array(X).astype("float32")
Y = np.array(Y).astype("float32")
Y = to_categorical(Y, num_classes)

X, X_test, Y, Y_test = train_test_split(X, Y, test_size=test_size, random_state = 42)
print(X.shape)
print(X_test.shape)
print(Y.shape)
print(Y_test.shape)
```

```
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
(1649, 100, 100)
(413, 100, 100)
(1649, 10)
(413, 10)
```

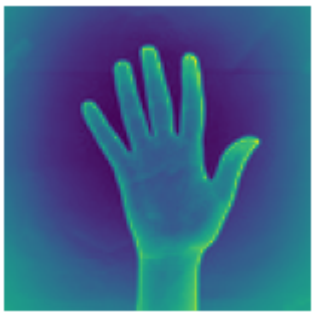
In [ ]:

```
img_size = 64

plt.subplot(1 , 2 , 1)
```

```
plt.imshow(X[0])
plt.axis("off")
```

Out[ ]: (-0.5, 99.5, 99.5, -0.5)



In [ ]:
```
from keras.utils.np_utils import to_categorical
## unroll the height and width and thickness into one big vector
x_train = X.reshape(1649, 10000)
x_test = X_test.reshape(413, 10000)
x_train = x_train.astype("float32")
x_test = x_test.astype("float32")

## normalize pixel values from 0 to 255
# data is already normalized
# x_train /= 255
# x_test /= 255


# y_train = to_categorical(Y, 10)
# Y_test = to_categorical(Y_test, 10)
y_train = Y
y_test = Y_test

print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(1649, 10000)
(1649, 10)
(413, 10000)
(413, 10)
```

Part A: The general idea is that I kept using neural network with one hidden layer and one dropout, with momentum stochastic gradient descent, different activation functions, batch size and epoch. I used activation function "sigmoid" at first, with batch size = 50. As the accuracy is pretty lower, at around 0.10, I changed parameters many times. For example, I changed batch size by increasing it gradually. As it still not working, I decided to use other activation function such as softmax. Softmax is used for normalizing the outputs, since it can convert them from weighted sum values into probabilities that sum to one. The value in the output of the softmax function will be interpreted as the prbability of membership for each class. And I also decreased the value of batch size. As a result, the accuracy increased to 0.40~0.50. The accuracy enhanced to the largest (0.7191) with batch size = 36 and activation function = sigmoid.

In [ ]:
```
model = Sequential()
model.add(Dense(10, activation='sigmoid', input_dim =x_train.shape[1]))
#One hidden layer + one dropout
```

```
model.add(Dropout(0.1))
model.add(Dense(5, activation='sigmoid'))
model.add(Dense(10, activation='sigmoid'))

model.summary()

#Momentum Stochastic Gradient Descent
sgd = keras.optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

#Compile the model
model.compile(loss='categorical_crossentropy',
              optimizer=sgd, metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 10)                100010

 dropout (Dropout)           (None, 10)                0

 dense_1 (Dense)             (None, 5)                 55

 dense_2 (Dense)             (None, 10)                60

=================================================================
Total params: 100,125
Trainable params: 100,125
Non-trainable params: 0
_________________________________________________________________
/usr/local/lib/python3.8/dist-packages/keras/optimizers/optimizer_v2/gradient_descent.p
y:108: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(SGD, self).__init__(name, **kwargs)
```

In [ ]:
```
model.fit(x_train, y_train,batch_size=32,
              epochs=100, verbose=2)
```

```
Epoch 1/100
52/52 - 1s - loss: 2.3173 - accuracy: 0.1079 - 758ms/epoch - 15ms/step
Epoch 2/100
52/52 - 0s - loss: 2.2974 - accuracy: 0.1249 - 179ms/epoch - 3ms/step
Epoch 3/100
52/52 - 0s - loss: 2.2865 - accuracy: 0.1395 - 159ms/epoch - 3ms/step
Epoch 4/100
52/52 - 0s - loss: 2.2759 - accuracy: 0.1589 - 156ms/epoch - 3ms/step
Epoch 5/100
52/52 - 0s - loss: 2.2624 - accuracy: 0.1953 - 166ms/epoch - 3ms/step
Epoch 6/100
52/52 - 0s - loss: 2.2444 - accuracy: 0.2116 - 169ms/epoch - 3ms/step
Epoch 7/100
52/52 - 0s - loss: 2.2238 - accuracy: 0.2256 - 151ms/epoch - 3ms/step
Epoch 8/100
52/52 - 0s - loss: 2.1953 - accuracy: 0.2438 - 164ms/epoch - 3ms/step
Epoch 9/100
52/52 - 0s - loss: 2.1607 - accuracy: 0.2771 - 164ms/epoch - 3ms/step
Epoch 10/100
52/52 - 0s - loss: 2.1240 - accuracy: 0.2577 - 158ms/epoch - 3ms/step
Epoch 11/100
52/52 - 0s - loss: 2.0832 - accuracy: 0.2735 - 153ms/epoch - 3ms/step
Epoch 12/100
52/52 - 0s - loss: 2.0333 - accuracy: 0.2996 - 154ms/epoch - 3ms/step
Epoch 13/100
52/52 - 0s - loss: 1.9854 - accuracy: 0.3184 - 154ms/epoch - 3ms/step
Epoch 14/100
```

```
52/52 - 0s - loss: 1.9273 - accuracy: 0.3354 - 153ms/epoch - 3ms/step
Epoch 15/100
52/52 - 0s - loss: 1.8778 - accuracy: 0.3517 - 160ms/epoch - 3ms/step
Epoch 16/100
52/52 - 0s - loss: 1.8299 - accuracy: 0.3493 - 156ms/epoch - 3ms/step
Epoch 17/100
52/52 - 0s - loss: 1.7756 - accuracy: 0.3954 - 162ms/epoch - 3ms/step
Epoch 18/100
52/52 - 0s - loss: 1.7421 - accuracy: 0.3863 - 166ms/epoch - 3ms/step
Epoch 19/100
52/52 - 0s - loss: 1.6905 - accuracy: 0.4148 - 158ms/epoch - 3ms/step
Epoch 20/100
52/52 - 0s - loss: 1.6585 - accuracy: 0.4196 - 157ms/epoch - 3ms/step
Epoch 21/100
52/52 - 0s - loss: 1.6349 - accuracy: 0.4196 - 165ms/epoch - 3ms/step
Epoch 22/100
52/52 - 0s - loss: 1.6012 - accuracy: 0.4263 - 163ms/epoch - 3ms/step
Epoch 23/100
52/52 - 0s - loss: 1.5675 - accuracy: 0.4524 - 171ms/epoch - 3ms/step
Epoch 24/100
52/52 - 0s - loss: 1.5340 - accuracy: 0.4700 - 147ms/epoch - 3ms/step
Epoch 25/100
52/52 - 0s - loss: 1.5117 - accuracy: 0.4633 - 154ms/epoch - 3ms/step
Epoch 26/100
52/52 - 0s - loss: 1.4811 - accuracy: 0.4839 - 154ms/epoch - 3ms/step
Epoch 27/100
52/52 - 0s - loss: 1.4638 - accuracy: 0.4779 - 168ms/epoch - 3ms/step
Epoch 28/100
52/52 - 0s - loss: 1.4406 - accuracy: 0.4773 - 158ms/epoch - 3ms/step
Epoch 29/100
52/52 - 0s - loss: 1.4189 - accuracy: 0.5070 - 160ms/epoch - 3ms/step
Epoch 30/100
52/52 - 0s - loss: 1.3991 - accuracy: 0.5112 - 160ms/epoch - 3ms/step
Epoch 31/100
52/52 - 0s - loss: 1.3761 - accuracy: 0.5215 - 157ms/epoch - 3ms/step
Epoch 32/100
52/52 - 0s - loss: 1.3520 - accuracy: 0.5294 - 153ms/epoch - 3ms/step
Epoch 33/100
52/52 - 0s - loss: 1.3268 - accuracy: 0.5434 - 162ms/epoch - 3ms/step
Epoch 34/100
52/52 - 0s - loss: 1.3239 - accuracy: 0.5549 - 158ms/epoch - 3ms/step
Epoch 35/100
52/52 - 0s - loss: 1.3045 - accuracy: 0.5555 - 152ms/epoch - 3ms/step
Epoch 36/100
52/52 - 0s - loss: 1.3024 - accuracy: 0.5488 - 157ms/epoch - 3ms/step
Epoch 37/100
52/52 - 0s - loss: 1.2761 - accuracy: 0.5531 - 151ms/epoch - 3ms/step
Epoch 38/100
52/52 - 0s - loss: 1.2440 - accuracy: 0.5858 - 163ms/epoch - 3ms/step
Epoch 39/100
52/52 - 0s - loss: 1.2452 - accuracy: 0.5585 - 149ms/epoch - 3ms/step
Epoch 40/100
52/52 - 0s - loss: 1.2239 - accuracy: 0.5797 - 165ms/epoch - 3ms/step
Epoch 41/100
52/52 - 0s - loss: 1.2108 - accuracy: 0.5797 - 156ms/epoch - 3ms/step
Epoch 42/100
52/52 - 0s - loss: 1.2081 - accuracy: 0.5858 - 175ms/epoch - 3ms/step
Epoch 43/100
52/52 - 0s - loss: 1.1770 - accuracy: 0.6064 - 150ms/epoch - 3ms/step
Epoch 44/100
52/52 - 0s - loss: 1.1684 - accuracy: 0.6113 - 152ms/epoch - 3ms/step
Epoch 45/100
52/52 - 0s - loss: 1.1617 - accuracy: 0.6173 - 153ms/epoch - 3ms/step
Epoch 46/100
52/52 - 0s - loss: 1.1252 - accuracy: 0.6173 - 164ms/epoch - 3ms/step
```

```
Epoch 47/100
52/52 - 0s - loss: 1.1066 - accuracy: 0.6374 - 153ms/epoch - 3ms/step
Epoch 48/100
52/52 - 0s - loss: 1.0940 - accuracy: 0.6446 - 156ms/epoch - 3ms/step
Epoch 49/100
52/52 - 0s - loss: 1.0733 - accuracy: 0.6531 - 152ms/epoch - 3ms/step
Epoch 50/100
52/52 - 0s - loss: 1.0784 - accuracy: 0.6489 - 154ms/epoch - 3ms/step
Epoch 51/100
52/52 - 0s - loss: 1.0640 - accuracy: 0.6458 - 153ms/epoch - 3ms/step
Epoch 52/100
52/52 - 0s - loss: 1.0644 - accuracy: 0.6604 - 156ms/epoch - 3ms/step
Epoch 53/100
52/52 - 0s - loss: 1.0206 - accuracy: 0.7047 - 168ms/epoch - 3ms/step
Epoch 54/100
52/52 - 0s - loss: 1.0138 - accuracy: 0.6810 - 150ms/epoch - 3ms/step
Epoch 55/100
52/52 - 0s - loss: 0.9997 - accuracy: 0.6828 - 156ms/epoch - 3ms/step
Epoch 56/100
52/52 - 0s - loss: 0.9942 - accuracy: 0.6895 - 154ms/epoch - 3ms/step
Epoch 57/100
52/52 - 0s - loss: 0.9785 - accuracy: 0.6931 - 153ms/epoch - 3ms/step
Epoch 58/100
52/52 - 0s - loss: 0.9469 - accuracy: 0.7132 - 155ms/epoch - 3ms/step
Epoch 59/100
52/52 - 0s - loss: 0.9393 - accuracy: 0.7095 - 159ms/epoch - 3ms/step
Epoch 60/100
52/52 - 0s - loss: 0.9294 - accuracy: 0.7216 - 155ms/epoch - 3ms/step
Epoch 61/100
52/52 - 0s - loss: 0.9265 - accuracy: 0.7144 - 155ms/epoch - 3ms/step
Epoch 62/100
52/52 - 0s - loss: 0.9013 - accuracy: 0.7271 - 151ms/epoch - 3ms/step
Epoch 63/100
52/52 - 0s - loss: 0.8980 - accuracy: 0.7314 - 151ms/epoch - 3ms/step
Epoch 64/100
52/52 - 0s - loss: 0.8922 - accuracy: 0.7338 - 156ms/epoch - 3ms/step
Epoch 65/100
52/52 - 0s - loss: 0.8475 - accuracy: 0.7526 - 170ms/epoch - 3ms/step
Epoch 66/100
52/52 - 0s - loss: 0.8636 - accuracy: 0.7508 - 157ms/epoch - 3ms/step
Epoch 67/100
52/52 - 0s - loss: 0.8575 - accuracy: 0.7489 - 157ms/epoch - 3ms/step
Epoch 68/100
52/52 - 0s - loss: 0.8267 - accuracy: 0.7641 - 153ms/epoch - 3ms/step
Epoch 69/100
52/52 - 0s - loss: 0.8414 - accuracy: 0.7532 - 150ms/epoch - 3ms/step
Epoch 70/100
52/52 - 0s - loss: 0.8070 - accuracy: 0.7629 - 154ms/epoch - 3ms/step
Epoch 71/100
52/52 - 0s - loss: 0.8034 - accuracy: 0.7635 - 168ms/epoch - 3ms/step
Epoch 72/100
52/52 - 0s - loss: 0.7898 - accuracy: 0.7659 - 153ms/epoch - 3ms/step
Epoch 73/100
52/52 - 0s - loss: 0.7920 - accuracy: 0.7774 - 157ms/epoch - 3ms/step
Epoch 74/100
52/52 - 0s - loss: 0.7694 - accuracy: 0.7799 - 155ms/epoch - 3ms/step
Epoch 75/100
52/52 - 0s - loss: 0.7616 - accuracy: 0.7768 - 154ms/epoch - 3ms/step
Epoch 76/100
52/52 - 0s - loss: 0.7780 - accuracy: 0.7744 - 155ms/epoch - 3ms/step
Epoch 77/100
52/52 - 0s - loss: 0.7603 - accuracy: 0.7714 - 161ms/epoch - 3ms/step
Epoch 78/100
52/52 - 0s - loss: 0.7457 - accuracy: 0.7908 - 166ms/epoch - 3ms/step
Epoch 79/100
```

```
52/52 - 0s - loss: 0.7247 - accuracy: 0.8047 - 155ms/epoch - 3ms/step
Epoch 80/100
52/52 - 0s - loss: 0.7224 - accuracy: 0.7944 - 159ms/epoch - 3ms/step
Epoch 81/100
52/52 - 0s - loss: 0.7362 - accuracy: 0.7702 - 156ms/epoch - 3ms/step
Epoch 82/100
52/52 - 0s - loss: 0.7066 - accuracy: 0.7962 - 160ms/epoch - 3ms/step
Epoch 83/100
52/52 - 0s - loss: 0.6860 - accuracy: 0.8059 - 158ms/epoch - 3ms/step
Epoch 84/100
52/52 - 0s - loss: 0.6883 - accuracy: 0.8084 - 167ms/epoch - 3ms/step
Epoch 85/100
52/52 - 0s - loss: 0.6924 - accuracy: 0.8041 - 156ms/epoch - 3ms/step
Epoch 86/100
52/52 - 0s - loss: 0.6919 - accuracy: 0.8035 - 158ms/epoch - 3ms/step
Epoch 87/100
52/52 - 0s - loss: 0.6621 - accuracy: 0.8035 - 161ms/epoch - 3ms/step
Epoch 88/100
52/52 - 0s - loss: 0.6529 - accuracy: 0.8187 - 154ms/epoch - 3ms/step
Epoch 89/100
52/52 - 0s - loss: 0.6694 - accuracy: 0.8102 - 158ms/epoch - 3ms/step
Epoch 90/100
52/52 - 0s - loss: 0.6456 - accuracy: 0.8150 - 169ms/epoch - 3ms/step
Epoch 91/100
52/52 - 0s - loss: 0.6237 - accuracy: 0.8381 - 154ms/epoch - 3ms/step
Epoch 92/100
52/52 - 0s - loss: 0.6367 - accuracy: 0.8181 - 155ms/epoch - 3ms/step
Epoch 93/100
52/52 - 0s - loss: 0.6130 - accuracy: 0.8326 - 155ms/epoch - 3ms/step
Epoch 94/100
52/52 - 0s - loss: 0.6305 - accuracy: 0.8241 - 155ms/epoch - 3ms/step
Epoch 95/100
52/52 - 0s - loss: 0.6210 - accuracy: 0.8223 - 158ms/epoch - 3ms/step
Epoch 96/100
52/52 - 0s - loss: 0.6135 - accuracy: 0.8229 - 155ms/epoch - 3ms/step
Epoch 97/100
52/52 - 0s - loss: 0.6035 - accuracy: 0.8308 - 168ms/epoch - 3ms/step
Epoch 98/100
52/52 - 0s - loss: 0.6152 - accuracy: 0.8193 - 160ms/epoch - 3ms/step
Epoch 99/100
52/52 - 0s - loss: 0.5721 - accuracy: 0.8532 - 158ms/epoch - 3ms/step
Epoch 100/100
52/52 - 0s - loss: 0.5979 - accuracy: 0.8266 - 156ms/epoch - 3ms/step
```

Out[ ]:
```
<keras.callbacks.History at 0x7fbe8e392520>
```

In [ ]:
```
#Evaluate with test data
model.evaluate(x_test, y_test, batch_size=32)
```

```
13/13 [==============================] - 0s 2ms/step - loss: 0.9987 - accuracy: 0.6780
```

Out[ ]:
```
[0.9987353086471558, 0.6779661178588867]
```

Part B: In this part, I changed the activation function to relu, and used Nesterov momentum stochastic gradient descent, dropouts, L2 regularization and random Gaussian weight initialization with 1/sqrt(n) standard deviation. Specifically, for layers, I created from 1 layers and kept adding to 3 layers. For epoch, the smallest I used is 5, and the largest I used is 100. By changing layers, dropouts, batch size, and epoch, I got the highest accuracy at around 0.20.

In [ ]:
```
#create a model structure, fit the model with train data, evaluate with test data
#Neural Network with three layers
model = Sequential()
model.add(Dense(32, activation='relu', input_dim =x_train.shape[1], kernel_regularizer=
```

```
                kernel_initializer=initializers.RandomNormal(mean=0,stddev=0.036)))
#hidden Layer + dropout
model.add(Dropout(0.1))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='relu'))
model.summary()

#Momentum Stochastic Gradient Descent
sgd = keras.optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

#Compile the model
model.compile(loss='categorical_crossentropy',
              optimizer=sgd, metrics=['accuracy'])
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 32)                320032

 dropout_1 (Dropout)         (None, 32)                0

 dense_4 (Dense)             (None, 32)                1056

 dense_5 (Dense)             (None, 10)                330

=================================================================
Total params: 321,418
Trainable params: 321,418
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
model.fit(x_train, y_train,batch_size=32,
                epochs = 100, verbose=2)
```

```
Epoch 1/100
52/52 - 1s - loss: 9.0494 - accuracy: 0.1055 - 738ms/epoch - 14ms/step
Epoch 2/100
52/52 - 0s - loss: 8.0545 - accuracy: 0.1061 - 218ms/epoch - 4ms/step
Epoch 3/100
52/52 - 0s - loss: 7.4265 - accuracy: 0.1061 - 210ms/epoch - 4ms/step
Epoch 4/100
52/52 - 0s - loss: 6.9296 - accuracy: 0.1061 - 225ms/epoch - 4ms/step
Epoch 5/100
52/52 - 0s - loss: 6.5296 - accuracy: 0.1031 - 205ms/epoch - 4ms/step
Epoch 6/100
52/52 - 0s - loss: 6.2059 - accuracy: 0.0995 - 225ms/epoch - 4ms/step
Epoch 7/100
52/52 - 0s - loss: 5.9445 - accuracy: 0.1037 - 212ms/epoch - 4ms/step
Epoch 8/100
52/52 - 0s - loss: 5.7332 - accuracy: 0.1061 - 208ms/epoch - 4ms/step
Epoch 9/100
52/52 - 0s - loss: 5.5616 - accuracy: 0.1079 - 211ms/epoch - 4ms/step
Epoch 10/100
52/52 - 0s - loss: 5.4216 - accuracy: 0.0988 - 214ms/epoch - 4ms/step
Epoch 11/100
52/52 - 0s - loss: 5.3092 - accuracy: 0.1092 - 222ms/epoch - 4ms/step
Epoch 12/100
52/52 - 0s - loss: 5.2183 - accuracy: 0.1164 - 208ms/epoch - 4ms/step
Epoch 13/100
52/52 - 0s - loss: 5.1390 - accuracy: 0.1383 - 219ms/epoch - 4ms/step
Epoch 14/100
52/52 - 0s - loss: 4.6605 - accuracy: 0.1322 - 210ms/epoch - 4ms/step
Epoch 15/100
```

```
52/52 - 0s - loss: 4.0226 - accuracy: 0.1031 - 241ms/epoch - 5ms/step
Epoch 16/100
52/52 - 0s - loss: 3.9240 - accuracy: 0.1164 - 212ms/epoch - 4ms/step
Epoch 17/100
52/52 - 0s - loss: 3.8475 - accuracy: 0.0988 - 210ms/epoch - 4ms/step
Epoch 18/100
52/52 - 0s - loss: 3.7888 - accuracy: 0.0946 - 230ms/epoch - 4ms/step
Epoch 19/100
52/52 - 0s - loss: 3.7397 - accuracy: 0.1025 - 212ms/epoch - 4ms/step
Epoch 20/100
52/52 - 0s - loss: 3.6982 - accuracy: 0.1043 - 224ms/epoch - 4ms/step
Epoch 21/100
52/52 - 0s - loss: 3.6671 - accuracy: 0.0970 - 207ms/epoch - 4ms/step
Epoch 22/100
52/52 - 0s - loss: 3.6385 - accuracy: 0.1098 - 213ms/epoch - 4ms/step
Epoch 23/100
52/52 - 0s - loss: 3.6182 - accuracy: 0.1025 - 207ms/epoch - 4ms/step
Epoch 24/100
52/52 - 0s - loss: 3.5975 - accuracy: 0.1128 - 211ms/epoch - 4ms/step
Epoch 25/100
52/52 - 0s - loss: 3.5801 - accuracy: 0.1140 - 222ms/epoch - 4ms/step
Epoch 26/100
52/52 - 0s - loss: 3.5598 - accuracy: 0.1267 - 224ms/epoch - 4ms/step
Epoch 27/100
52/52 - 0s - loss: 3.5327 - accuracy: 0.1304 - 211ms/epoch - 4ms/step
Epoch 28/100
52/52 - 0s - loss: 2.8226 - accuracy: 0.1383 - 211ms/epoch - 4ms/step
Epoch 29/100
52/52 - 0s - loss: 2.5897 - accuracy: 0.1013 - 230ms/epoch - 4ms/step
Epoch 30/100
52/52 - 0s - loss: 2.5479 - accuracy: 0.1007 - 203ms/epoch - 4ms/step
Epoch 31/100
52/52 - 0s - loss: 2.4744 - accuracy: 0.0946 - 218ms/epoch - 4ms/step
Epoch 32/100
52/52 - 0s - loss: 2.4416 - accuracy: 0.1037 - 206ms/epoch - 4ms/step
Epoch 33/100
52/52 - 0s - loss: 2.4163 - accuracy: 0.1001 - 211ms/epoch - 4ms/step
Epoch 34/100
52/52 - 0s - loss: 2.3950 - accuracy: 0.1055 - 219ms/epoch - 4ms/step
Epoch 35/100
52/52 - 0s - loss: 2.3763 - accuracy: 0.1007 - 209ms/epoch - 4ms/step
Epoch 36/100
52/52 - 0s - loss: 2.3633 - accuracy: 0.0934 - 214ms/epoch - 4ms/step
Epoch 37/100
52/52 - 0s - loss: 2.3523 - accuracy: 0.0849 - 205ms/epoch - 4ms/step
Epoch 38/100
52/52 - 0s - loss: 2.3431 - accuracy: 0.1067 - 215ms/epoch - 4ms/step
Epoch 39/100
52/52 - 0s - loss: 2.3353 - accuracy: 0.1067 - 224ms/epoch - 4ms/step
Epoch 40/100
52/52 - 0s - loss: 2.3301 - accuracy: 0.0885 - 212ms/epoch - 4ms/step
Epoch 41/100
52/52 - 0s - loss: 2.3249 - accuracy: 0.0946 - 212ms/epoch - 4ms/step
Epoch 42/100
52/52 - 0s - loss: 2.3212 - accuracy: 0.0958 - 219ms/epoch - 4ms/step
Epoch 43/100
52/52 - 0s - loss: 2.3164 - accuracy: 0.0964 - 222ms/epoch - 4ms/step
Epoch 44/100
52/52 - 0s - loss: 2.3165 - accuracy: 0.1031 - 213ms/epoch - 4ms/step
Epoch 45/100
52/52 - 0s - loss: 2.3120 - accuracy: 0.1031 - 213ms/epoch - 4ms/step
Epoch 46/100
52/52 - 0s - loss: 2.3119 - accuracy: 0.0928 - 204ms/epoch - 4ms/step
Epoch 47/100
52/52 - 0s - loss: 2.3114 - accuracy: 0.0946 - 222ms/epoch - 4ms/step
```

```
Epoch 48/100
52/52 - 0s - loss: 2.3083 - accuracy: 0.1007 - 219ms/epoch - 4ms/step
Epoch 49/100
52/52 - 0s - loss: 2.3082 - accuracy: 0.1061 - 218ms/epoch - 4ms/step
Epoch 50/100
52/52 - 0s - loss: 2.3068 - accuracy: 0.0982 - 212ms/epoch - 4ms/step
Epoch 51/100
52/52 - 0s - loss: 2.3070 - accuracy: 0.0946 - 230ms/epoch - 4ms/step
Epoch 52/100
52/52 - 0s - loss: 2.3063 - accuracy: 0.1061 - 213ms/epoch - 4ms/step
Epoch 53/100
52/52 - 0s - loss: 2.3054 - accuracy: 0.1055 - 216ms/epoch - 4ms/step
Epoch 54/100
52/52 - 0s - loss: 2.3047 - accuracy: 0.0988 - 212ms/epoch - 4ms/step
Epoch 55/100
52/52 - 0s - loss: 2.3045 - accuracy: 0.0964 - 214ms/epoch - 4ms/step
Epoch 56/100
52/52 - 0s - loss: 2.3045 - accuracy: 0.0982 - 204ms/epoch - 4ms/step
Epoch 57/100
52/52 - 0s - loss: 2.3035 - accuracy: 0.0928 - 230ms/epoch - 4ms/step
Epoch 58/100
52/52 - 0s - loss: 2.3058 - accuracy: 0.0976 - 211ms/epoch - 4ms/step
Epoch 59/100
52/52 - 0s - loss: 2.3043 - accuracy: 0.1019 - 216ms/epoch - 4ms/step
Epoch 60/100
52/52 - 0s - loss: 2.3038 - accuracy: 0.1007 - 207ms/epoch - 4ms/step
Epoch 61/100
52/52 - 0s - loss: 2.3022 - accuracy: 0.1043 - 216ms/epoch - 4ms/step
Epoch 62/100
52/52 - 0s - loss: 2.2992 - accuracy: 0.1237 - 214ms/epoch - 4ms/step
Epoch 63/100
52/52 - 0s - loss: 2.2844 - accuracy: 0.1486 - 207ms/epoch - 4ms/step
Epoch 64/100
52/52 - 0s - loss: 2.2488 - accuracy: 0.1649 - 220ms/epoch - 4ms/step
Epoch 65/100
52/52 - 0s - loss: 2.2353 - accuracy: 0.1534 - 211ms/epoch - 4ms/step
Epoch 66/100
52/52 - 0s - loss: 2.2412 - accuracy: 0.1516 - 207ms/epoch - 4ms/step
Epoch 67/100
52/52 - 0s - loss: 2.2036 - accuracy: 0.1722 - 227ms/epoch - 4ms/step
Epoch 68/100
52/52 - 0s - loss: 2.3390 - accuracy: 0.1001 - 211ms/epoch - 4ms/step
Epoch 69/100
52/52 - 0s - loss: 2.3214 - accuracy: 0.0849 - 220ms/epoch - 4ms/step
Epoch 70/100
52/52 - 0s - loss: 2.3181 - accuracy: 0.1001 - 208ms/epoch - 4ms/step
Epoch 71/100
52/52 - 0s - loss: 2.3144 - accuracy: 0.1019 - 217ms/epoch - 4ms/step
Epoch 72/100
52/52 - 0s - loss: 2.3127 - accuracy: 0.1061 - 210ms/epoch - 4ms/step
Epoch 73/100
52/52 - 0s - loss: 2.3111 - accuracy: 0.0995 - 217ms/epoch - 4ms/step
Epoch 74/100
52/52 - 0s - loss: 2.3095 - accuracy: 0.1061 - 208ms/epoch - 4ms/step
Epoch 75/100
52/52 - 0s - loss: 2.3093 - accuracy: 0.0964 - 217ms/epoch - 4ms/step
Epoch 76/100
52/52 - 0s - loss: 2.3081 - accuracy: 0.0898 - 221ms/epoch - 4ms/step
Epoch 77/100
52/52 - 0s - loss: 2.3074 - accuracy: 0.0988 - 214ms/epoch - 4ms/step
Epoch 78/100
52/52 - 0s - loss: 2.3068 - accuracy: 0.0982 - 215ms/epoch - 4ms/step
Epoch 79/100
52/52 - 0s - loss: 2.3058 - accuracy: 0.0922 - 224ms/epoch - 4ms/step
Epoch 80/100
```

```
52/52 - 0s - loss: 2.3059 - accuracy: 0.1061 - 207ms/epoch - 4ms/step
Epoch 81/100
52/52 - 0s - loss: 2.3051 - accuracy: 0.0885 - 221ms/epoch - 4ms/step
Epoch 82/100
52/52 - 0s - loss: 2.3049 - accuracy: 0.0976 - 222ms/epoch - 4ms/step
Epoch 83/100
52/52 - 0s - loss: 2.3053 - accuracy: 0.0940 - 219ms/epoch - 4ms/step
Epoch 84/100
52/52 - 0s - loss: 2.3055 - accuracy: 0.0916 - 212ms/epoch - 4ms/step
Epoch 85/100
52/52 - 0s - loss: 2.3050 - accuracy: 0.1061 - 219ms/epoch - 4ms/step
Epoch 86/100
52/52 - 0s - loss: 2.3043 - accuracy: 0.0940 - 209ms/epoch - 4ms/step
Epoch 87/100
52/52 - 0s - loss: 2.3048 - accuracy: 0.1061 - 223ms/epoch - 4ms/step
Epoch 88/100
52/52 - 0s - loss: 2.3045 - accuracy: 0.0879 - 216ms/epoch - 4ms/step
Epoch 89/100
52/52 - 0s - loss: 2.3045 - accuracy: 0.0922 - 215ms/epoch - 4ms/step
Epoch 90/100
52/52 - 0s - loss: 2.3045 - accuracy: 0.1049 - 219ms/epoch - 4ms/step
Epoch 91/100
52/52 - 0s - loss: 2.3051 - accuracy: 0.0964 - 212ms/epoch - 4ms/step
Epoch 92/100
52/52 - 0s - loss: 2.3069 - accuracy: 0.0946 - 211ms/epoch - 4ms/step
Epoch 93/100
52/52 - 0s - loss: 2.3060 - accuracy: 0.1061 - 214ms/epoch - 4ms/step
Epoch 94/100
52/52 - 0s - loss: 2.3055 - accuracy: 0.0946 - 210ms/epoch - 4ms/step
Epoch 95/100
52/52 - 0s - loss: 2.3053 - accuracy: 0.0970 - 215ms/epoch - 4ms/step
Epoch 96/100
52/52 - 0s - loss: 2.3050 - accuracy: 0.1061 - 216ms/epoch - 4ms/step
Epoch 97/100
52/52 - 0s - loss: 2.3049 - accuracy: 0.0976 - 207ms/epoch - 4ms/step
Epoch 98/100
52/52 - 0s - loss: 2.3048 - accuracy: 0.0952 - 210ms/epoch - 4ms/step
Epoch 99/100
52/52 - 0s - loss: 2.3048 - accuracy: 0.1001 - 218ms/epoch - 4ms/step
Epoch 100/100
52/52 - 0s - loss: 2.3045 - accuracy: 0.0946 - 214ms/epoch - 4ms/step
```

Out[ ]:
```
<keras.callbacks.History at 0x7fbe8e21d9d0>
```

In [ ]:
```
#Evaluate with test data
model.evaluate(x_test, y_test, batch_size=32)
```

```
13/13 [==============================] - 0s 4ms/step - loss: 2.3092 - accuracy: 0.0775
```

Out[ ]:
```
[2.3091976642608643, 0.07748184353113174]
```

Part C: In this part, I used tuner to help me generate the best network model. However, the highest accuracy is at around 0.30, which is smaller than the accuracy in Part A.

In [ ]:
```
!pip install keras-tuner --upgrade
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
ic/simple/
Requirement already satisfied: keras-tuner in /usr/local/lib/python3.8/dist-packages (1.
1.3)
Requirement already satisfied: ipython in /usr/local/lib/python3.8/dist-packages (from k
eras-tuner) (7.9.0)
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from
keras-tuner) (2.23.0)
Requirement already satisfied: tensorboard in /usr/local/lib/python3.8/dist-packages (fr
```

```
om keras-tuner) (2.9.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from ker
as-tuner) (1.21.6)
Requirement already satisfied: packaging in /usr/local/lib/python3.8/dist-packages (from
keras-tuner) (21.3)
Requirement already satisfied: kt-legacy in /usr/local/lib/python3.8/dist-packages (from
keras-tuner) (1.0.4)
Requirement already satisfied: prompt-toolkit<2.1.0,>=2.0.0 in /usr/local/lib/python3.8/
dist-packages (from ipython->keras-tuner) (2.0.10)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.8/dist-packages
(from ipython->keras-tuner) (5.6.0)
Requirement already satisfied: jedi>=0.10 in /usr/local/lib/python3.8/dist-packages (fro
m ipython->keras-tuner) (0.18.2)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.8/dist-package
s (from ipython->keras-tuner) (57.4.0)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.8/dist-packages (fr
om ipython->keras-tuner) (0.7.5)
Requirement already satisfied: backcall in /usr/local/lib/python3.8/dist-packages (from
ipython->keras-tuner) (0.2.0)
Requirement already satisfied: pygments in /usr/local/lib/python3.8/dist-packages (from
ipython->keras-tuner) (2.6.1)
Requirement already satisfied: decorator in /usr/local/lib/python3.8/dist-packages (from
ipython->keras-tuner) (4.4.2)
Requirement already satisfied: pexpect in /usr/local/lib/python3.8/dist-packages (from i
python->keras-tuner) (4.8.0)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in /usr/local/lib/python3.8/dist-pack
ages (from jedi>=0.10->ipython->keras-tuner) (0.8.3)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.8/dist-packages (from p
rompt-toolkit<2.1.0,>=2.0.0->ipython->keras-tuner) (0.2.5)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.8/dist-packages (fro
m prompt-toolkit<2.1.0,>=2.0.0->ipython->keras-tuner) (1.15.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.8/dist
-packages (from packaging->keras-tuner) (3.0.9)
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.8/dist-packages
(from pexpect->ipython->keras-tuner) (0.7.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-packag
es (from requests->keras-tuner) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/li
b/python3.8/dist-packages (from requests->keras-tuner) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packa
ges (from requests->keras-tuner) (2022.9.24)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (f
rom requests->keras-tuner) (2.10)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.8/dist-pa
ckages (from tensorboard->keras-tuner) (3.19.6)
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.8/dist-packages (fr
om tensorboard->keras-tuner) (0.38.4)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python
3.8/dist-packages (from tensorboard->keras-tuner) (0.4.6)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.8/dist-pa
ckages (from tensorboard->keras-tuner) (2.15.0)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.8/dist-packages
(from tensorboard->keras-tuner) (1.0.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/p
ython3.8/dist-packages (from tensorboard->keras-tuner) (0.6.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.
8/dist-packages (from tensorboard->keras-tuner) (1.8.1)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.8/dist-packages
(from tensorboard->keras-tuner) (3.4.1)
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.8/dist-packages (f
rom tensorboard->keras-tuner) (1.3.0)
Requirement already satisfied: grpcio>=1.24.3 in /usr/local/lib/python3.8/dist-packages
(from tensorboard->keras-tuner) (1.51.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.8/dist-pa
ckages (from google-auth<3,>=1.6.3->tensorboard->keras-tuner) (0.2.8)
```

```
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.8/dist-p
ackages (from google-auth<3,>=1.6.3->tensorboard->keras-tuner) (5.2.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.8/dist-packages
(from google-auth<3,>=1.6.3->tensorboard->keras-tuner) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.8/dist
-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard->keras-tuner) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.8/dist-
packages (from markdown>=2.6.8->tensorboard->keras-tuner) (4.13.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.8/dist-packages (from
importlib-metadata>=4.4->markdown>=2.6.8->tensorboard->keras-tuner) (3.11.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.8/dist-pac
kages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard->keras-tuner) (0.
4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.8/dist-packages
(from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard->keras-tun
er) (3.2.2)
```

In [ ]:
```
import keras_tuner
from tensorflow import keras
from tensorflow.keras import layers
```

In [ ]:
```
# Neural Network with two hidden layers
def build_model(hp):
    # hyperparam we want to tune: number of neurons, lr, activation func
    activation_func = hp.Choice("activation", ["sigmoid","relu"])
    learning_rate = hp.Float("lr", min_value=1e-3, max_value=1e-1, sampling="log")
    neuron_num = hp.Int("neuron-1", min_value=32, max_value=128, step=32)

    # create a model with two hidden layers
    model = Sequential()
    model.add((Dense(32, activation='sigmoid', input_dim =x_train.shape[1])))
    model.add(Dense(units=neuron_num,
                    activation=activation_func))
    model.add(Dense(units=neuron_num,
                    activation=activation_func))
    model.add(Dense(10, activation=activation_func))

    sgd = keras.optimizers.SGD(lr=learning_rate, decay=1e-6, momentum=0.9, nesterov=Tru

    #Compile the model
    model.compile(loss='categorical_crossentropy',
                optimizer=sgd, metrics=['accuracy'])
    return model

build_model(keras_tuner.HyperParameters())
tuner = keras_tuner.RandomSearch(
    build_model,
    objective='val_loss',
    max_trials=10)
```

In [ ]:
```
tuner.search(x_train, y_train, epochs=5, validation_data=(x_test, y_test))
best_model = tuner.get_best_models()[0]
```

```
WARNING:tensorflow:Detecting that an object or model or tf.train.Checkpoint is being del
eted with unrestored values. See the following logs for the specific values in question.
To silence these warnings, use `status.expect_partial()`. See https://www.tensorflow.or
g/api_docs/python/tf/train/Checkpoint#restorefor details about the status object returne
d by the restore function.
WARNING:tensorflow:Value in checkpoint could not be found in the restored object: (roo
t).layer_with_weights-3.kernel
WARNING:tensorflow:Value in checkpoint could not be found in the restored object: (roo
```

```
t).layer_with_weights-3.bias
WARNING:tensorflow:Value in checkpoint could not be found in the restored object: (roo
t).optimizer.iter
WARNING:tensorflow:Value in checkpoint could not be found in the restored object: (roo
t).optimizer.decay
WARNING:tensorflow:Value in checkpoint could not be found in the restored object: (roo
t).optimizer.learning_rate
WARNING:tensorflow:Value in checkpoint could not be found in the restored object: (roo
t).optimizer.momentum
WARNING:tensorflow:Value in checkpoint could not be found in the restored object: (roo
t).optimizer's state 'momentum' for (root).layer_with_weights-3.kernel
WARNING:tensorflow:Value in checkpoint could not be found in the restored object: (roo
t).optimizer's state 'momentum' for (root).layer_with_weights-3.bias
```

In [ ]:
```
best_model.evaluate(x_test, y_test)
```

```
13/13 [==============================] - 0s 3ms/step - loss: 1.9399 - accuracy: 0.3002
```

Out[ ]:
```
[1.939896821975708, 0.3002421259880066]
```

In [ ]:
```
tuner.results_summary()
```

```
Results summary
Results in ./untitled_project
Showing 10 best trials
<keras_tuner.engine.objective.Objective object at 0x7fcbfa2bc5b0>
Trial summary
Hyperparameters:
activation: sigmoid
lr: 0.0633606054822938
neuron-1: 96
Score: 1.939896821975708
Trial summary
Hyperparameters:
activation: sigmoid
lr: 0.0031531212959390142
neuron-1: 64
Score: 2.3012585639953613
Trial summary
Hyperparameters:
activation: sigmoid
lr: 0.00198317724026896
neuron-1: 64
Score: 2.305431842803955
Trial summary
Hyperparameters:
activation: sigmoid
lr: 0.007116103500829736
neuron-1: 96
Score: 2.3058016300201416
Trial summary
Hyperparameters:
activation: sigmoid
lr: 0.0012066632140502118
neuron-1: 64
Score: 2.307534694671631
Trial summary
Hyperparameters:
activation: relu
lr: 0.012341315610031297
neuron-1: 32
Score: 3.4939591884613037
Trial summary
Hyperparameters:
activation: relu
```

```
lr: 0.0062658328965630354
neuron-1: 64
Score: 4.135528564453125
Trial summary
Hyperparameters:
activation: relu
lr: 0.0028925868757120306
neuron-1: 64
Score: 5.072015762329102
Trial summary
Hyperparameters:
activation: relu
lr: 0.018977192017024146
neuron-1: 64
Score: 5.930873870849609
Trial summary
Hyperparameters:
activation: relu
lr: 0.07006893747238556
neuron-1: 32
Score: 7.032344341278076
```

In [ ]:
```
best_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                320032

 dense_1 (Dense)             (None, 96)                3168

 dense_2 (Dense)             (None, 96)                9312

 dense_3 (Dense)             (None, 10)                970

=================================================================
Total params: 333,482
Trainable params: 333,482
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
tuner.search_space_summary()
```

```
Search space summary
Default search space size: 3
activation (Choice)
{'default': 'sigmoid', 'conditions': [], 'values': ['sigmoid', 'relu'], 'ordered': Fals
e}
lr (Float)
{'default': 0.001, 'conditions': [], 'min_value': 0.001, 'max_value': 0.1, 'step': None,
'sampling': 'log'}
neuron-1 (Int)
{'default': None, 'conditions': [], 'min_value': 32, 'max_value': 128, 'step': 32, 'samp
ling': None}
```